ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени федеральное государственное

Бюджетное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра информатики

Лабораторная работа №2

По дисциплине: «Технология разработки программного обеспечения»

Тема: «Основы объектно-ориентированногопрограммирования»

Выполнил студент:

Группы: БФИ1902

Шацкий Е. И.

Проверила:

Мосева. М.С.

Москва, 2020 г.

**Содержание**

[1 Задачи на лабораторную работу. 3](#_Toc52969210)

[2 Ход решения лабораторной работы. 3](#_Toc52969211)

[Список используемых источников 7](#_Toc52969212)

# **1 Задачи на лабораторную работу.**

Для выполнения лабораторной работы необходимо:

а) Обеспечить реализацию метода equals () в файле Location.java;

б) Обеспечить реализацию метода hashcode() в файле Location.java;

в) Создать метод numOpenWaypoints() в файле AStarState.java;

г) Создать метод getMinOpenWaypoint() в файле AStarState.java;

д) Создать метод addOpenWaypoint(Waypoint newWP) в файле AStarState.java;

е) Создать метод isLocationClosed(Location loc) в файле AStarState.java;

ж) Создать метод closeWaypoint(Location loc) в файле AStarState.java;

# **2 Ход решения лабораторной работы.**

После написания алгоритма работы программы необходимо ее запустить. Этот процесс показан на рисунке 1.

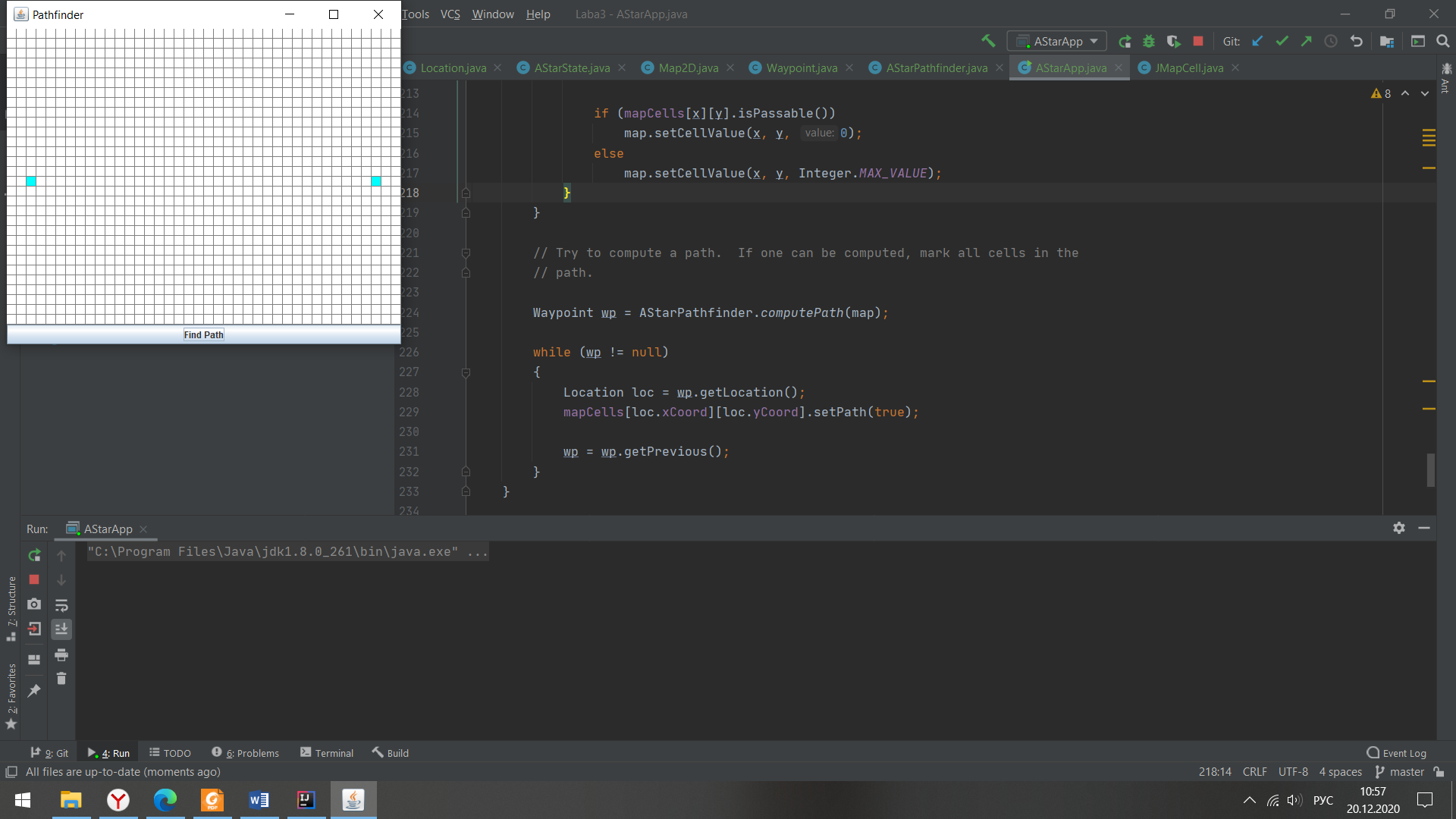


Рисунок 1 –Запуск программы.

На листинге 1 представлен код файла Location.java. На листинге 2 представлен код AStarState.java.

Листинг 1 - Код файла Location.java;

import java.util.Objects;  
  
/\*\*  
 \* This class represents a specific location in a 2D map. Coordinates are  
 \* integer values.  
 \*\*/  
public class Location  
{  
 /\*\* X coordinate of this location. \*\*/  
 public int xCoord;  
  
 /\*\* Y coordinate of this location. \*\*/  
 public int yCoord;  
  
  
 /\*\* Creates a new location with the specified integer coordinates. \*\*/  
 public Location(int x, int y)  
 {  
 xCoord = x;  
 yCoord = y;  
 }  
  
 /\*\* Creates a new location with coordinates (0, 0). \*\*/  
 public Location()  
 {  
 this(0, 0);  
 }  
  
 @Override  
 public boolean equals(Object obj)  
 {  
 if (obj instanceof Location)  
 {  
 Location loc = (Location) obj;  
 if (xCoord == loc.xCoord &&  
 yCoord == loc.yCoord)  
 {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(xCoord, yCoord);  
 }  
  
}

Листинг 2 - Код файла AStarState.java;

import java.util.HashMap;  
  
/\*\*  
 \* This class stores the basic state necessary for the A\* algorithm to compute a  
 \* path across a map. This state includes a collection of "open waypoints" and  
 \* another collection of "closed waypoints." In addition, this class provides  
 \* the basic operations that the A\* pathfinding algorithm needs to perform its  
 \* processing.  
 \*\*/  
public class AStarState  
{  
 /\*\* This is a reference to the map that the A\* algorithm is navigating. \*\*/  
 private Map2D map;  
  
 private HashMap<Location, Waypoint> openWaypoints = new HashMap<Location, Waypoint>();  
  
 private HashMap<Location, Waypoint> closedWaypoints = new HashMap<Location, Waypoint>();  
  
 /\*\*  
 \* Initialize a new state object for the A\* pathfinding algorithm to use.  
 \*\*/  
 public AStarState(Map2D map)  
 {  
 if (map == null)  
 throw new NullPointerException("map cannot be null");  
  
 this.map = map;  
 }  
  
 /\*\* Returns the map that the A\* pathfinder is navigating. \*\*/  
 public Map2D getMap()  
 {  
 return map;  
 }  
  
 /\*\*  
 \* This method scans through all open waypoints, and returns the waypoint  
 \* with the minimum total cost. If there are no open waypoints, this method  
 \* returns <code>null</code>.  
 \*\*/  
 public Waypoint getMinOpenWaypoint()  
 {  
 Waypoint point = null;  
 double min = 10000;  
 double resultWaypoint = 0;  
 for (Waypoint i : openWaypoints.values()){  
 resultWaypoint = i.getTotalCost();  
 if(resultWaypoint < min){  
 min = resultWaypoint;  
 point = i;  
 }  
 }  
  
 return point;  
 }  
  
 /\*\*  
 \* This method adds a waypoint to (or potentially updates a waypoint already  
 \* in) the "open waypoints" collection. If there is not already an open  
 \* waypoint at the new waypoint's location then the new waypoint is simply  
 \* added to the collection. However, if there is already a waypoint at the  
 \* new waypoint's location, the new waypoint replaces the old one <em>only  
 \* if</em> the new waypoint's "previous cost" value is less than the current  
 \* waypoint's "previous cost" value.  
 \*\*/  
 public boolean addOpenWaypoint(Waypoint newWP)  
 {  
 Waypoint shortPoint = openWaypoints.get(newWP.loc);  
 if(shortPoint == null || newWP.getPreviousCost() < shortPoint.getPreviousCost()){  
 openWaypoints.put(newWP.loc, newWP);  
 return true;  
 }  
 return false;  
 }  
  
  
 /\*\* Returns the current number of open waypoints. \*\*/  
 public int numOpenWaypoints()  
 {  
 return openWaypoints.size();  
 }  
  
  
 /\*\*  
 \* This method moves the waypoint at the specified location from the  
 \* open list to the closed list.  
 \*\*/  
 public void closeWaypoint(Location loc)  
 {  
 Waypoint delPoint = openWaypoints.remove(loc);  
 closedWaypoints.put(loc, delPoint);  
 }  
  
 /\*\*  
 \* Returns true if the collection of closed waypoints contains a waypoint  
 \* for the specified location.  
 \*\*/  
 public boolean isLocationClosed(Location loc)  
 {  
 return closedWaypoints.containsKey(loc);  
 }  
}

На рисунке 2 представлен результат работы программы.

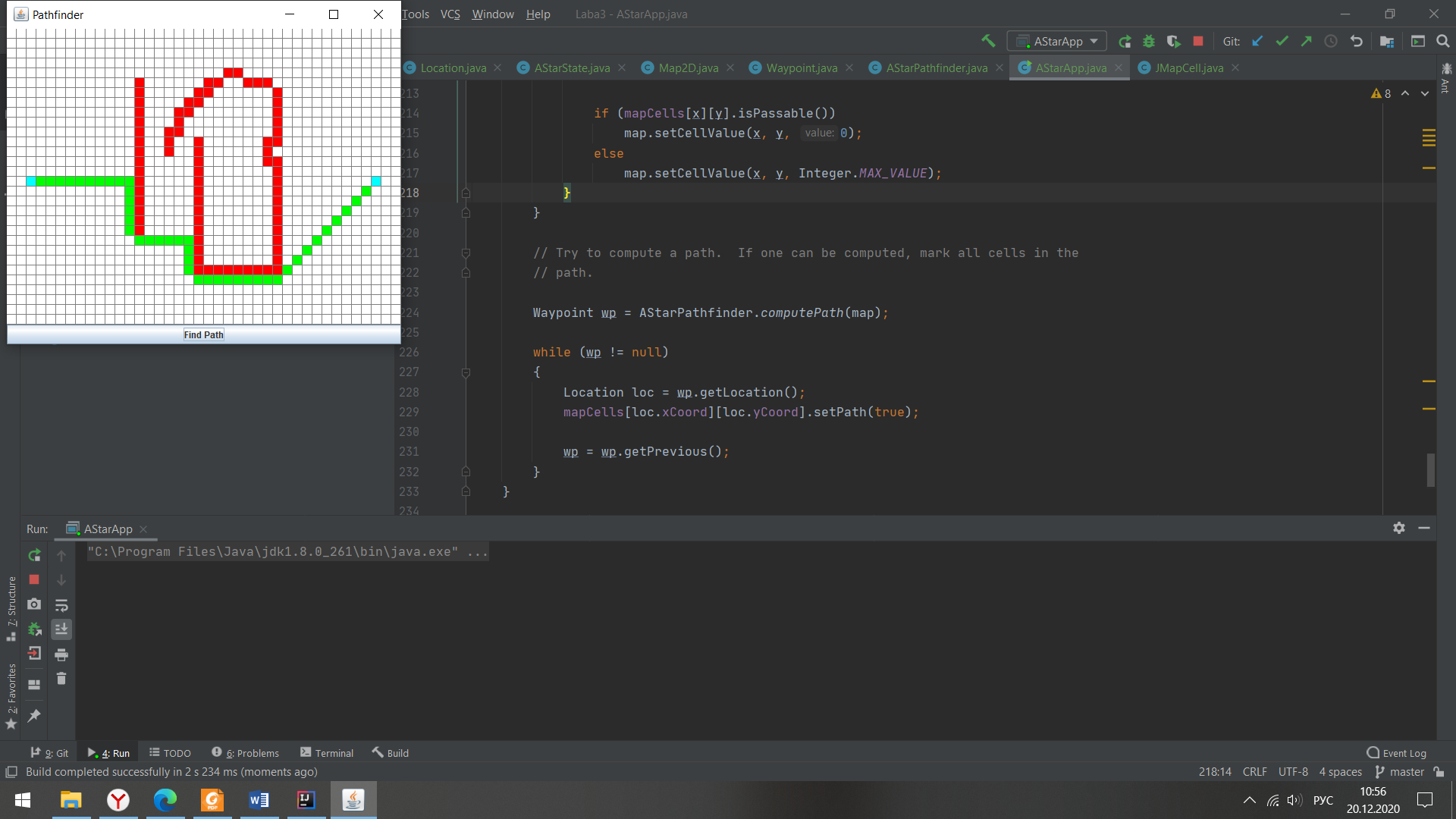


Рисунок 2 – Результат работы программы.

# **Список используемых источников**

1.ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчёт о научно-исследовательской работе. Структура и правила оформления.

2.ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления.